BSFP: Blockchain-Enabled Smart Parking with Fairness, Reliability and Privacy Protection

Can Zhang, Liehuang Zhu, *Member, IEEE,* Chang Xu, Chuan Zhang, Kashif Sharif, *Member, IEEE*, Huishu Wu, Hannes Westermann

***Abstract*—The convenience of using private cars has an accompanying parking challenge which becomes a significant issue in congested metropolitans and downtown areas. The explosive increase in the number of vehicles has substantially raised the issue of finding a suitable parking spot, which is both time and resource consuming. At the same time, many private parking spots remain idle, while their owners are not present at home. To promote the utility of private parking spots and mitigate parking issues, smart parking apps can be used. Unfortunately, some of them suffer from privacy issues that affect participation willingness, while others work in a centralized environment where the availability of service is not guaranteed in the presence of malicious users. In this work, we propose Blockchain-based Smart parking with Fairness, reliability and Privacy protection, called BSFP. Specifically, group signatures, bloom filters, and vector-based encryption are leveraged to protect the user's privacy. The decentralized nature of blockchain is utilized to achieve reliability in smart parking, and the smart contract is used to realize fairness. Comprehensive security analysis and experimental results based on the real-world dataset show that BSFP achieves fairness, reliability and privacy protection with high efficiency.**

***Index Terms*—Vehicular Network, Blockchain, Smart Parking, Privacy Protection, Fairness.**

I. Introduction

THE advancement in the comfort level and affordability of vehicular technology has encouraged more people to buy vehicles for the convenience of transportation. However, the large number of cars has led to parking issues which have been a major problem for congested metropolitan areas. Statistics of 2017 [1], shows that Beijing needs around 1.3 million more parking spots, and the estimated number can reach 50 million in China. Therefore, many drivers have to drive around while searching for a suitable parking space, and this behavior is known as cruising. The process of cruising for parking spots is not only time-consuming but also results in significant resource wastage. Another study [2] shows that 30% of traffic congestion originates from drivers searching for parking spots, and the average search time is 8.1 min. It also shows that such processes result in 945,000 additional extra miles, 47,000 extra burned gasoline, and 728 tons of carbon dioxide emissions in Los Angeles, over a year.

On the other hand, there exist numerous idle private parking spots. Most of these parking spots are in private spaces such as residential areas, and the utility of these is relatively low. For instance, a parking owner working away from home for 8 hours per day, has the parking spot free for that duration of time. Hence, if these idle parking spots can be effectively utilized, the problem of parking issues will be significantly reduced. At the same time, this can also result in financial benefit for the owners to offset some of their maintenance and paring costs.

In recent times, some smartphone apps have been providing parking spot sharing facilities between drivers and parking owners. Although this solution is quite simple and effective, yet it introduces significant privacy issues. If the owner's parking spot information and the driver's parking request are uploaded to the cloud server without any privacy protection, both the untrustworthy server and malicious adversaries can infer sensitive information (e.g., home/work address, favorite place to go, health condition, and real-world identity) with the help of background knowledge. These privacy issues limit the willingness of both parking owners and drivers to embrace the convenience and resource-saving offered by modern technology.

The research interest in privacy concerns (e.g., access control [3]–[5], and cloud security [6]–[9]) has garnered significant attention from the community. Hence, a diverse number of privacy-preserving mechanisms have been presented, which cover multiple applications in vehicular networks (e.g., autonomous driving [10], vehicular crowdsensing [11]–[13], access control [14], [15], vehicular cloud computing [16], [17], and vehicular social network [18], [19]). To tackle the privacy issues of smart parking, some privacy-preserving smart parking schemes [20]–[23] have been proposed to hide the real identity or exact location for both drivers and parking owners. However, these existing schemes cannot directly used in the smart parking scenario mentioned above. Besides, some of them utilize a centralized cloud server to store the parking spot information and match the parking spot for drivers or deploy distributed RSUs (Road-Side Units) to manage and relay the parking information. However, several challenges still need attention in preserving the privacy of different entities in the system.

This work is supported by Key Research and Development Program for Guangdong Province under Grant No. 2019B010137003, the National Natural Science Foundation of China (Grant Nos. 61972037, U1836212), the Prestudy Foundation of Weapons and Equipment (No. 3151102401), and grant No. 2019CX10014 from the Graduate Technological Innovation Project of Beijing Institute of Technology.

Can Zhang, Liehuang Zhu (Corresponding Author), Chang Xu (Corresponding Author), Chuan Zhang, and Kashif Sharif are with the School of Computer Science and Technology at Beijing Institute of Technology, Beijing, China (E-mail: {canzhang, liehuangz, xuchang, chuanz, 7620160009}@bit.edu.cn). Huishu Wu and Hannes Westermann major in innovation, science, technology and law at University of Montreal, Montreal, Canada (E-mail: wuhuishu0122@gmail.com, hannes.westermann@umontreal.ca).

In a centralized server design, large bandwidth and computational requirements are needed for the system to work efficiently. If the server fails then the system availability cannot be guaranteed. In addition, the server might return incorrect matching results due to random failures or attacks by adversaries. In a distributed RSU model, individual units cannot have large storage and computation capacities like centralized cloud servers or nodes, hence it cannot undertake the task of information storage and parking spot matching operation. Besides, they require inter-communication among them to ensure system-wide coverage however, the low bandwidth in VANET cannot undertake high communications which may cause network congestion. Therefore, both the centralized server and the distribute RSU models cannot guarantee the reliability of large-scale smart parking services.

In addition to the above limitations, most of the proposed schemes do not consider fairness, especially when some malicious users disobey the parking rules or intentionally try to game the system. For example, a driver may refuse to pay parking fees to the parking owner, or some parking owners may publish fake parking spot information to generate more revenue. In addition, when interacting with the parking service provider, both drivers and parking owners need to pay service charges. However, as discussed above, the reliability of service cannot be guaranteed. In general, the users may not be able to get the services for which they have paid the corresponding fees, in existing smart parking schemes. Hence, fairness cannot be guaranteed.

In light of these shortcomings of existing schemes, we propose a decentralized privacy-preserving smart parking scheme (BSFP), that achieves both fairness and reliability. To the best of our knowledge, this is the first decentralized and privacy-preserving smart parking scheme that satisfies both fairness and reliability.

Following are the major contributions of this work:

- We propose a decentralized structure for smart parking by leveraging the blockchain, which eliminates the issues brought from the centralized server and distributed RSUs, and provides reliable parking services.
- We present BSFP, the blockchain-based smart parking with fairness, reliability and privacy protection. In BSFP, the BBS group signature is used to realize anonymous authentication for both drivers and parking owners, while providing the traceability of malicious users. Secure pseudo-random functions and bloom filters are used to achieve privacy-preserving location-based range queries. Vector-based encryption is used to enable privacy-preserving time matching between the driver and the parking spot. Note that the proposed privacy-preserving location-based range query and time matching can be used not only in our proposed scheme but also in other relevant application scenarios.
- We design a smart contract for blockchain to achieve fairness. The driver can obtain the correct parking match results if they pay. The parking owner can publish their parking spot information correctly, and receive the corresponding money paid by the driver who rents the parking spot. Also, the honest workers in the blockchain can get rewards while the misbehaving users will be punished.
- We present a thorough security analysis and comprehensive experimental evaluation based on a real-world dataset to show that BSFP achieves privacy protection, data integrity, reliability, authentication, and traceability with high computation & communication efficiency.

The rest of this paper is organized as follows. In Section II we describe the related works about smart parking, blockchain usage in the vehicular network, and financial fairness. We make a brief introduction of the blockchain, BBS signature and privacy-preserving range query in Section III. Section IV presents the formal system model, threat model, and design goals. The detailed construction of BSFP and the design of a smart contract is introduced in Section V and Section VI, respectively. Security analysis is given in Section VII, and experimental results are presented in Section VIII. Finally, Section IX concludes this paper.

II. Related Works

A. Smart Parking

The parking problem is considered an important issue as it has attracted the attention and interest of many researchers. Lin et al. [24] presented a survey of smart parking solutions covering 2000-2016. The authors discussed the basic issues in smart parking and gave valuable suggestions for research directions and commercial products.

Although some smart parking mechanisms [25], [26] try to improve the utility and service quality, they do not consider the privacy issues that exist in smart parking. In recent years, a series of privacy-preserving smart parking [20]–[23] have been proposed to protect identity privacy and location privacy. Lu et al. [20] proposed an intelligent, secure and privacy-preserving parking scheme that utilizes RSUs to manage large parking lots without sacrificing the driver's privacy. Ni et al. [21] presented P-SPAN, an efficient and privacy-preserving smart parking navigation system. Compared with the existing schemes, P-SPAN achieves efficient navigation result retrieval with the help of Bloom Filters. Zhu et al. [22] presented ASAP, which achieves both anonymous smart parking and anonymous payment. In ASAP, a centralized cloud server is used to store all locations as a hashmap, and a short randomizable signature with E-cash is utilized to achieve anonymous parking and payment.

Recently, Amiri et al. [23] presented a blockchain-based privacy-preserving smart parking system that makes use of Private Information Retrieval (PIR) and short randomizable signature to protect the driver's location and identity privacy. Unfortunately, this system only considers the parking scenario based on public parking lots instead of private parking spots. Besides, it cannot achieve fairness and traceability as well.

Some of the proposed schemes cannot fit in our smart parking scenario where private idle parking spots are owned by parking owners. Some of the existing schemes discussed above use a centralized structure that cannot counter the misbehavior of malicious central nodes, while others use distributed RSUs that cannot undertake high computation and communication overloads, thus jeopardizing their reliability. Besides, they do not focus on the fairness of the smart parking solution.

B. Blockchain Meets Vehicular Networks

Blockchain has become popular in recent years due to the properties of decentralization and immutability. Bitcoin [27] first used blockchain as an underlying platform to create a decentralized, reliable and trustless payment system. Vehicular network, as a part of the space-air-ground integrated network (SAGIN) [28]–[30], has also caught the attention of academic and industry circles [31]. Now blockchain has been integrated into various domains [32] including vehicular networks. Gao et al. [33] proposed a blockchain-based payment mechanism for Vehicle-to-Grid (V2G) networks. To solve the conflicts between privacy protection and data sharing in V2G networks, they designed a new transaction structure with the corresponding verification algorithm to guarantee reliability and scalability. Sharma et al. [34] introduced B2VDM, a decentralized architecture for vehicular data management in a vehicular network. B2VDM makes use of blockchain to achieve load distribution and consensus among RSUs, which maintains the system reliability. Li et al. [35] designed a blockchain-assisted efficient and privacy-preserving carpooling scheme. They constructed a private blockchain where RSUs are considered as decentralized blockchain nodes and designed a transaction structure closely related to the proposed carpooling scheme. Detailed analysis shows that it achieves privacy protection with high efficiency.

However, to the best of our knowledge, none of the existing schemes can be directly used to solve the issues of smart parking. This forms the basic motivation of our work.

C. Enabling Financial Fairness

Bitcoin uses its native cryptocurrency BTC to continuously motivate honest miners to generate the correct block in the Bitcoin network. Hence, it can be utilized to realize fair protocols [36]–[38]. Recent work on *financial fairness* [39] argue that financial fairness means that the misbehaving party will be financially penalized while the remaining honest parties will receive a corresponding compensation. Li et al. [40] proposed CreditCoin, an incentive mechanism to encourage honest users to share traffic information while tracing the malicious users to penalize them. Hu et al. [41] presented a privacy-preserving keyword search by leveraging smart contracts. The scheme achieves fairness in both single-user settings and multi-user settings. Their definition of fairness includes: 1) As long as the data owner pays searching fees to workers (i.e., the blockchain miners), it will receive the correct search results. 2) As long as the miner honestly follows the protocol, they will earn money. 3) In a multi-user setting, in addition to these two properties mentioned above, other users will receive correct search results as long as they pay to both workers and data owners. The data owner earns money as long as it gives the search token to users.

Inspired by the existing works that mentioned above, we propose to achieve fairness in blockchain-enabled smart parking by financially rewarding well-behaved users while misbehaved users will be penalized.

III. Preliminaries

In this section, we present the basic information related to blockchain and smart contracts, BBS group signature [42], privacy-preserving range query [43], and other cryptographic primitives used in the proposed scheme.

A. Blockchain and Smart Contracts

Blockchain technology utilizes a distributed ledger to achieve reliable, trustless and decentralized transactions in a multi-party system. The two major properties of the blockchain are *decentralization* and *immutability*. Decentralization means that there is no centralized authority (or server) to manage the system. Moreover, blockchain cannot be controlled by a single enterprise or government. Immutability means that once the data is stored in the blockchain, it cannot be modified or deleted. These two properties make blockchain an append-only ledger with multiple replicas.

In a blockchain, all the blocks are linked in a sequence. Each block consists of a block header that stores the block information such as block number and previous block hash, and a block body that contains the transactions. Each transaction also has a corresponding transaction head that stores the transaction information such as transaction hash, transaction fee, and a body that stores the customized data or the data used by the smart contract.

To realize flexible and Turing-complete programmability of the blockchain, Ethereum [44] uses smart contracts. A smart contract is a computerized transaction protocol that executes the terms of a contract [45] automatically. In the blockchain, a smart contract is the chain code that is triggered by the users when they initiate a transaction.

B. BBS Group Signature

The BBS group signature [42] is based on a Strong Diffle-Hellman (SDH) assumption, and consists of following four algorithms which are described as follows:

- $(mpk, msk, SK) \leftarrow$ KeyGen(λ, n): A probabilistic algorithm that receives a security parameter λ and the number of group members n as input, and it outputs a public key mpk, a master private key msk, and a set SK with n private keys $sk_1, \cdots, sk_n$ for n members.
- $\sigma \leftarrow$ Sign(mpk, sk, m): A probabilistic algorithm that receives a public key mpk, a member's private key sk and a message m as input, and it outputs a signature σ.
- $\{true, false\} \leftarrow$ Verify(mpk, σ, m): A deterministic algorithm that receives a public key mpk, a signature σ and a message m as input, and it outputs $true$ if σ is valid, otherwise outputs $false$.
- $A_i \leftarrow$ Open(mpk, msk, σ_i, m_i): A deterministic algorithm that receives a public key mpk, a master private key msk, a signature σ_i and a message m_i as input, and it outputs an indicator A_i that can be used to find the identity of the signer i.

C. Privacy-Preserving Range Query

A privacy-preserving range query [43] consists of five processes: *Prefix Encoding*, *PBTree Construction*, *Node Randomization*, *Trapdoor Computation*, and *Query*, which are described as follows:

- **Prefix Encoding**: Given a binary string $b_1b_2\dots b_w$, the data owner calculates its prefix family $\mathcal{P}(b)$ which consists of $w+1$ prefixes $\{b_1b_2\dots b_w, \cdots, b_1*\dots*, *\dots*\}$; given a range $[a,b]$ represented by binary format, the user calculates a minimum set of prefixes $\mathcal{PM}(a,b)$.
- **PBTree Construction**: Given m prefix families, the data owner constructs a balanced binary tree PBTree pbt, where all leaf nodes are linked as a linked list.
- **Node Randomization**: r keys $k_1, \cdots, k_r$ are shared between the data owner and the user. For each prefix p_i in the node in pbt, the data owner calculates r double-hashes by generating HMAC(c, HMAC(k_j, p_i)) for each k_j, where c is a random number, and then inputs them in a Bloom Filter. Lastly, the data owner sends the encrypted pbt and data to the cloud server.
- **Trapdoor Computation**: Given a query range $[lb, rb]$ with a minimal prefix set $\mathcal{PM}(lb, rb)$, assume $\mathcal{PM}(lb, rb)$ has z prefixes. For each prefix p_i, the user calculates r hashes HMAC(k_1, p_i), $\cdots$, HMAC(k_r, p_i). Finally, a $z \times r$ matrix M is generated as the query trapdoor, which will be sent to the cloud server.
- **Query**: Given the query trapdoor M, the cloud server checks whether there exists a row i for every column j in $M[i]$, that the results of HMAC($c, M[i][j]$) in the Bloom Filter are all 1. Finally, the queried result will be returned to the user.

D. Cryptographic Primitives

In the proposed scheme, we use several cryptographic primitives to achieve privacy protection and data integrity. Two secure one-way pseudo-random functions F, G and a secure hash function H as described in Eq. 1:

$$F : \{0,1\}^\lambda \times \{0,1\}^* \to \{0,1\}^\lambda \tag{1a}$$
$$G : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^\lambda \tag{1b}$$
$$H : \{0,1\}^* \to \{0,1\}^\lambda \tag{1c}$$

where λ is the security parameter.

A secure pseudo-random function means that any Probabilistic Polynomial Time (PPT) adversary cannot distinguish it from the random functions. The one-way-ness of a function $f(x)$ means that it is easy to compute $f(x)$ when knowing x, however, x cannot be recovered by $f(x)$. A secure hash function h achieves one-way-ness as mentioned above, and collision-resistance which means that is impossible for any PPT adversary to find different x_1 and x_2 that satisfy $h(x_1) = h(x_2)$. The formal definition of secure pseudo-random function and secure hash function can be found in [46], hence we omit them for simplicity.

IV. Models & Problem Statement

Here, we first present the system model followed by the threat model and design goals of the proposed privacy-preserving, reliable and fair smart parking scheme.

A. System Model

The proposed system comprises of five main entities as shown in Fig. 1: *Trust Authority (TA)*, *Parking Owner*, *Driver*, *RSU* and *Blockchain Network*.

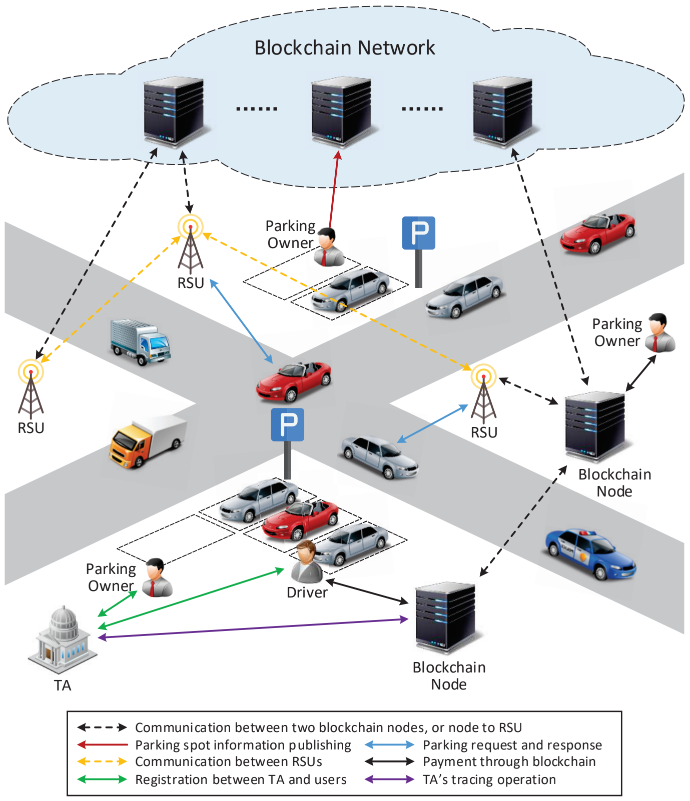

Blockchain Network
......
......
Parking Owner
RSU
RSU
RSU
Parking Owner
Blockchain Node
Parking Owner
Driver
TA
Blockchain Node
Communication between two blockchain nodes, or node to RSU
Parking spot information publishing
Communication between RSUs
Registration between TA and users
Parking request and response
Payment through blockchain
TA's tracing operation

Fig. 1: System model.

1) TA: In the system initialization phase, TA generates public parameters and several keys for each registered Parking Owner and Driver. After successful initialization, TA may remain offline unless any dispute occurs. Under these circumstances, the TA can trace the problem source and find the real-world identity of the misbehaving driver(s) or parking owner(s). Therefore, the introduction of TA does not affect the decentralization property.

2) Parking Owner: When the parking spot is not in use, its owner can rent it to generate profit. Every parking owner needs to use their real-world identity to register before forwarding parking spot information to the system. When a driver parks in his parking spot, he will get the parking fees paid by the driver.

3) Driver: We assume that each driver has an Electric Vehicle (EV) as a light-node of blockchain that can send and receive transactions. When a driver wants to park his vehicle somewhere, he will send a parking request to a nearby RSU. After receiving the search result, the driver will send a parking request to the parking owner associated with the search result. Finally, he drives to the parking location after contacting with the parking owner.

4) RSU: In our system, RSU plays an important role as an intermediary between drivers and blockchain network. When receiving a driver's parking request, it will authenticate the request. If the authentication is successful, it will re-encrypt the encrypted time vector, generate a new parking request and forward it to the blockchain node. Besides, it will transmit the parking spot matching results to the driver.

5) Blockchain: In the proposed system, blockchain can reliably store the encrypted parking requests (as transactions). In addition, the smart contract can handle the parking request, and it returns the parking spot matching results. The Blockchain network also provides a platform for anonymous and fair parking payment.

B. Threat Model

The TA is considered as trustworthy in our threat model, and cannot be compromised. It uses secure channels to distribute the secret keys to every driver and parking owner.

The model considers the RSUs to be honest-but-curious, which means that they will follow our proposed protocol and forward all the network packages. However, they may try to infer the driver's and parking owner's private information by the received packages.

Most drivers and parking owners are honest and will send requests and responses truthfully. However, there may exist several malicious drivers and parking owners. Malicious drivers can send a large number of fake parking requests to sabotage the system, like a DDoS attack [47]. They can also follow rule-breaking behaviors (e.g., the actual parking time is different from the demanded time). For malicious parking owners, they can publish fake parking spot information for profits.

Blockchain network is reliable, which means that adversaries cannot control more than 50% network power to compromise the integrity of consensus [48]. However, some malicious workers (i.e., blockchain miners) try to generate fake blocks to receive illegal profits. Also, because of the transparency of blockchain, adversaries can perform an inference attack to make statistical analysis on blockchain data to obtain sensitive information of drivers and parking owners.

Moreover, unauthorized (unregistered) users try to impersonate authorized (registered) users to publish parking spot information or send parking requests.

Note that collusion between users and RSUs, and collusion between users and workers are not allowed. These assumptions are reasonable because the privacy of corrupted users will also be revealed to others which harm their profits.

C. Design Goals

To achieve fairness, reliable and privacy-preserving smart parking, the solution should fulfill some fundamental requirements. The detailed design goals corresponding to these requirements lists as follows:

1) Privacy Protection: The driver's and parking owner's privacy should be preserved. The different types of privacy include *identity privacy* (i.e., driver's identity and parking owner's identity), *location privacy* (i.e., the queried location for drivers and the exact parking spot location for parking owners), *time privacy* (i.e., the available time for each parking spot and the parking time for each driver), and *data privacy* (i.e., sensitive information stored on blockchain). Neither the honest-but-curious RSUs nor malicious adversaries should be able to violate any type of privacy.

2) Integrity & Reliability: The proposed scheme should maintain data integrity during transmission and provide reliable storage. The malicious data modification during data transmission should be detected by the receiver. Besides, the decentralized network cannot be compromised, and the stored data cannot be tampered with by adversaries.

3) Authentication & Traceability: The drivers and parking owners should be authenticated to prevent unauthorized user from entering the system and impersonating legitimate users to interact with others. When some dispute occurs, TA needs to trace the identity of related users and realize the responsibility confirmation.

4) Fairness: The fairness of the proposed scheme includes two aspects: fair payment and reward & punishment. More specifically, to achieve fairness:

- The proposed scheme should guarantee fair payment for both drivers and parking owners. For each driver, as long as they send parking requests and pay for it, they will receive the correct matching results. For each parking owner, as long as a driver occupies their parking spot, they will gain appropriate payments by the driver as a reward.
- The honest worker participating in the decentralized network (i.e., the blockchain miner) should be rewarded, while misbehaving or malicious users and workers should be punished. The fines can be used to compensate the drivers and parking owners who suffer the losses.

5) Efficiency: The limited bandwidth and computational resources of VANET should be considered, and the communication and computation costs between drivers and RSUs should be low to realize fast response time and to enable efficient smart parking services.

V. Construction of BSFP

The proposed Blockchain-based Smart and Fair Parking (BSFP) scheme is a reliable and privacy-protecting solution. It consists of four phases: system initialization, parking spot publishing, parking request generation, and parking spot matching. The notations used in this work are shown in Table I. Each phase is explained in detail below.

A. System Initialization

In this phrase, TA first generates a shared key $K_0 \leftarrow \{0,1\}^\lambda$, and a shared key set $K = \{K_1, \cdots, K_o\}$ (for each $k \in K$, $k \leftarrow \{0,1\}^\lambda$), where λ is the security parameter. Next TA generates two invertible square matrices M_1, M_2 of order l.

TABLE I: List of Notations.

Notation	Description
M	The number of partitioned areas in a city.
m, n	The number of parking owners and drivers.
F, G	Secure pseudo-random functions.
H	Secure hash function.
K_0	A shared key for Area No. obfuscation.
$K = \{K_1, \cdots, K_o\}$	o shared keys for location-based range query.
$M_1^{l\times l}, M_2^{l\times l}, M_3^{l\times l}$	$l \times l$ matrices for time vector encryption.
λ	Security parameter.
P_i, D_j	Parking owner i and driver j.
Ppk	Public key for parking owners.
Psk_i	Private key for P_i.
Dpk	Public key for drivers.
Dsk_j	Private key for D_j.
(pk_R, sk_R)	Key-pair for RSU R.
A, A'	Plain & obfuscated Area No. in the city.
(x_i, y_i)	The exact parking spot location of P_i.
(L_x^i, L_y^i)	P_i's grid coordinates.
$(\mathcal{B}_x^i, \mathcal{B}_y^i, r_x^i, r_y^i)$	P_i's encrypted grid coordinates.
α_i, α_i'	P_i's plain & encrypted available time vector.
$\|\|$	String concatenation operator.
t_θ	Request time threshold.
$price_i$	P_i's parking price per hour.
σ	Digital signature.
$(gx_1^j, gx_2^j, gy_1^j, gy_2^j)$	Driver j's queried grid position range.
$(\mathcal{T}_x^j, \mathcal{T}_y^j)$	Driver j's location-based range query token.
β_j, β_j'	D_j's plain & encrypted parking time vector.
$\mathcal{C}$	The smart contract used in the proposed system.
$\mathcal{R}_j$	D_j's parking request.
$\mathcal{L}_i$	P_i's parking spot information.
$\mathcal{P}(b)$	A binary string b's prefix family [43].
$\mathcal{P}(a, b)$	The minimal prefix set [43] of range $[a, b]$.
BBS	BBS group signature scheme [42].
$\mathsf{addr_i}$	P_i's blockchain address.
$\mathsf{addr_j}$	D_j's blockchain address.

For m registered parking owners, TA executes $(Ppk, Pmsk, Psk) \leftarrow$ BBS.KeyGen(λ, m) to generate a public key Ppk, a master private key $Pmsk$, and m private keys $Psk = \{Psk_1, \cdots, Psk_m\}$. For n registered drivers, TA executes $(Dpk, Dmsk, Dsk) \leftarrow$ BBS.KeyGen(λ, n) to generate a public key Dpk, a master private key $Dmsk$, and n private keys $Dsk = \{Dsk_1, \cdots, Dsk_n\}$. For RSU R (we assume that there exists only one RSU for simplicity), TA generates a key-pair (pk_R, sk_R) and calculates $M_3 = M_1^{-1} \cdot M_2^{-1}$. Note that TA can use any public-key cryptosystem that supports digital signature (e.g., RSA) to generate R's key-pair.

Then TA sends $(K_0, K, M_1, Ppk, Dpk, Psk_i)$ to each parking owner i, sends $(K_0, K, M_2, Ppk, Dpk, Dsk_j)$ to each driver j, and sends (pk_R, sk_R, M_3) to R through secure channels.

Note that every registered parking owner and driver needs to submit their real-world identity to TA so that TA can trace the misbehaving and malicious users when any dispute occurs. We omit the detailed description of identity registration, as it is a one-time trivial process.

B. Parking Spot Publishing

In this phase, parking owners can publish the parking spot information to the blockchain to generate revenue from their idle parking spots. To improve efficiency, we first partition the whole city into M square neighborhood areas. Then, we partition each area into a grid pattern, as illustrated in Fig. 2. Here, the neighborhood has 12 grid coordinates, and each grid can be represented by a horizontal coordinate and a vertical coordinate. For instance, in Fig. 2, there exist a car and a parking spot, and their grid coordinates are $(1, 1)$ and $(4, 3)$, respectively.

For parking owner P_i, to protect the location privacy, it first converts its exact parking spot location (x_i, y_i) (which can be represented by longitude and latitude) to a grid coordinate (L_x^i, L_y^i) associated with an Area No. in the city noted as A_i. To obfuscate A_i, P_i uses K_0 to compute $A_i' = G(K_0, A_i)$. To encrypt the horizontal grid coordinate L_x^i, P_i first computes its binary prefix family $\mathcal{P}(L_x^i)$. Next, P_i chooses a random number r_x^i and an empty Bloom Filter $\mathcal{B}_x^i$. For each prefix $p \in \mathcal{P}(L_x^i)$, P_i calculates o λ-length bit strings $f_1, \cdots, f_o$ where $f_k = F(r_x^i, G(K_k, p))$, and inserts them to $\mathcal{B}_x^i$. After inserting all computed strings of all prefixes, P_i can obtain $(\mathcal{B}_x^i, r_x^i)$ as the encrypted horizontal grid coordinate. The encryption of vertical grid coordinate L_y^i is similar to what P_i does in encrypting L_x^i. After encryption, P_i can obtain $(\mathcal{B}_y^i, r_y^i)$ as the encrypted result.

To protect the time privacy, we encode the available time of parking spot as a time vector. The time vector $\alpha = \{a_1, \cdots, a_l\}$ consists of l dimensions and each dimension represents $\frac{24}{l}$ hours. For example, when $l = 48$, a_1 represents the time interval from 00:00 to 00:30, a_2 represents 00:30 - 01:00 and so on. If $a_t > 0$, it indicates that this parking spot is unavailable at this interval, and if $a_t = 0$, the interval of this parking spot is available. P_i uses this rule to encode the available time and creates a time vector α_i. Then P_i uses key M_1 to encrypt α_i and obtains the encrypted available time vector $\alpha_i' = \alpha_i \cdot M_1$.

Moreover, to communicate with and receive payments from drivers, the parking owner P_i needs to generate a blockchain address $\mathsf{addr_i}$ to send and receive blockchain transaction. Note that $\mathsf{addr_i}$ can be generated by P_i himself, and the generation does not use any real identity of P_i, which achieves account anonymity.

Finally, P_i uses $\mathsf{addr_i}$ to publish the encrypted parking spot information $\mathcal{L}_i = (A_i', \mathcal{B}_x^i, \mathcal{B}_y^i, r_x^i, r_y^i, \alpha_i', \mathsf{addr_i}, price_i, t_i, \sigma_i)$ to the blockchain by invoking the smart contract, where t_i represents the current timestamp that will be used for time verification, and $\sigma_i =$ BBS.Sign$(Ppk, Psk_i, H(tmp))$ is the BBS group signature signed by P_i, where $tmp = A_i'||\mathcal{B}_x^i||\mathcal{B}_y^i||r_x^i||r_y^i||\alpha_i'||\mathsf{addr_i}||price_i||t_i$.

When receiving $\mathcal{L}_i$, the smart contract $\mathcal{C}$ first makes a time verification by comparing $t - t_i$ and t_θ, where t stands for the current timestamp, t_i stands for P_i's publishing time in $\mathcal{L}_i$, and t_θ represents the time threshold. If $t - t_i > t_\theta$, $\mathcal{L}_i$ will be dropped as obsolete information. If the verification is successful, $\mathcal{C}$ verifies the BBS signature by executing BBS.Verify$(Ppk, \sigma_i, H(tmp))$, where $tmp = A_i'||\mathcal{B}_x^i||\mathcal{B}_y^i||r_x^i||r_y^i||\alpha_i'||\mathsf{addr_i}||price_i||t_i$. If the verification returns $true$, $\mathcal{C}$ will add (or update the historical) $\mathcal{L}_i$ to the blockchain, otherwise $\mathcal{L}_i$ will be rejected.

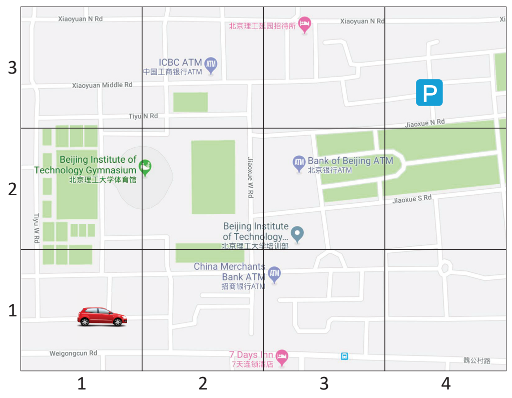


Fig. 2: An example of grid coordinates.

C. Parking Request Generation

When the driver D_j intends to park their car, they send an encrypted parking request to the blockchain through RSU. To generate parking request, D_j first designates a location range represented by $(gx_1^j, gx_2^j, gy_1^j, gy_2^j)$, where the horizontal grid coordinate ranges from $[gx_1^j, gx_2^j]$ and the vertical grid coordinate ranges from $[gy_1^j, gy_2^j]$.

D_j first obfuscates the located Area No. A_j similar to what the parking owner P_i does, as $A'_j = G(K_0, A_j)$. To generate the horizontal grid range query token, D_j first computes the minimum prefix set $\mathcal{P}(gx_1^j, gx_2^j)$. Assume that $\mathcal{P}(gx_1^j, gx_2^j)$ has z elements $p_1, \cdots, p_z$, D_j obtains the token by computing $\mathcal{T}_x^j = \bigcup_{i'=1}^{z}\{\bigcup_{k=1}^{o} G(K_k, p_{i'})\}$. The generation of vertical grid range query token is similar to what D_j does in generating $\mathcal{T}_x^j$. After generation, D_j obtains $\mathcal{T}_y^j$ as the vertical grid range search token.

Next, D_j encodes the parking time using a similar process as that of parking owner. The only difference is: $a_t > 0$ (or $a_t = 0$) indicates that D_j will (or will not) park at this time interval. The encoded time vector is noted as β_j. Then D_j uses the key M_2 to encrypt β_j and obtains the encrypted parking time vector $\beta'_j = M_2 \cdot \beta_j^T$.

Finally, similar to the parking owners, the driver D_i uses his blockchain address $\mathsf{addr_j}$ to invoke the smart contract and send the parking request $\mathcal{R}_j = (A'_j, \mathcal{T}_x^j, \mathcal{T}_y^j, \beta'_j, t_j, \sigma_j)$ to the blockchain through the RSU nearby, where t_j represents the current timestamp that will be used for time verification, and $\sigma_j = \text{BBS.Sign}(Dpk, Dsk_j, H(tmp))$ is the BBS group signature signed by D_j, where $tmp = A'_j||\mathcal{T}_x^j||\mathcal{T}_y^j||\beta'_j||t_j$.

When receiving $\mathcal{R}_j$, RSU first makes a time verification. If $t - t_j > t_\theta$, $\mathcal{R}_j$ will be dropped as an obsolete request. If the verification is successful, RSU then verifies the BBS signature by executing BBS.Verify$(Dpk, \sigma_j, H(tmp))$, where $tmp = A'_j||\mathcal{T}_x^j||\mathcal{T}_y^j||\beta'_j||t_j$. If the verification returns $true$, RSU will calculate $\beta''_j = M_3 \cdot \beta'_j = M_1^{-1} \cdot M_2^{-1} \cdot M_2 \cdot \beta_j^T = M_1^{-1} \cdot \beta_j^T$, and use the private key sk_R to generate a signature σ_R of $H(A'_j||\mathcal{T}_x^j||\mathcal{T}_y^j||\beta'_j||t_j||\sigma_j||\beta''_j)$. Then R generates a modified request $\mathcal{R}'_j = (A'_j, \mathcal{T}_x^j, \mathcal{T}_y^j, \beta'_j, t_j, \sigma_j, \beta''_j, \sigma_R)$ and sends it to blockchain. If the verification is failed, $\mathcal{R}_j$ will be rejected.

D. Parking Spot Matching

When receiving D_j's parking request $\mathcal{R}'_j$ generated by RSU R, the peer of the blockchain network first uses R's public key pk_R to verify the signature σ_R. If the verification is successful, it will invoke the associated smart contract. The smart contract $\mathcal{C}$ makes a parking spot matching that depends on the partitioned area, the encrypted grid location range and encrypted parking time. This phase includes two steps: *location-based matching* and *time-based matching*.

1) Location-based Matching: The smart contract $\mathcal{C}$ first parses the grid range search token $(\mathcal{T}_x^j, \mathcal{T}_y^j)$. Then for each published parking spot information $\mathcal{L}_i$ that satisfies $A'_i == A'_j$ (which means that the driver associated with Area No. A'_i and the parking spots associated with A'_j belong to the same area), it checks whether $\mathcal{L}_i$'s grid coordinate is in the queried grid range. To check the horizontal grid coordinate, $\mathcal{C}$ first gets the Bloom Filter $\mathcal{B}_x^i$ and random number r_x^i from $\mathcal{L}_i$, and parses the token $\mathcal{T}_x^j$ into z groups $\mathcal{T}_1, \cdots, \mathcal{T}_z$, where each $\mathcal{T}_{i'}$ consists of o λ-length strings $g_1, \cdots, g_o$. For each $\mathcal{T}_{i'}$, $\mathcal{C}$ calculates $\mathcal{T}'_{i'} = \bigcup_{k=1}^{o} F(r_i^x, g_k)$ and queries each $f \in \mathcal{T}'_{i'}$ into $\mathcal{B}_x^i$. If there exists a $\mathcal{T}'_{i'}$ that for all $f \in \mathcal{T}'_{i'}$, the query returns *true*, and $\mathcal{C}$ will then check the vertical grid coordinate. Otherwise, it means $\mathcal{L}_i$ is not in the queried location range, and $\mathcal{C}$ will check the next parking spot information that is stored on blockchain. The vertical grid coordinate checking is similar to what $\mathcal{C}$ does in checking the horizontal grid coordinate. If the checking is successful, $\mathcal{L}_i$ will be added to a candidate set S. Finally, $\mathcal{C}$ outputs S as the grid range query result, which will be used next.

2) Time-based matching: After the location-based matching, $\mathcal{C}$ will perform the time-based matching using the following steps: For each $\mathcal{L}_i \in S$, $\mathcal{C}$ executes $\alpha'_i \cdot \beta'_j = (\alpha_i \cdot M_1) \cdot (M_1^{-1} \cdot \beta_j^T) = \alpha_i \cdot \beta_j^T$, where α'_i and β'_j represent $\mathcal{L}_i$'s encrypted available time vector and D_j's parking time vector, respectively. If $\alpha'_i \cdot \beta'_j \neq 0$, $\mathcal{L}_i$ will be filtered because the time is not matched. Otherwise, $\mathcal{L}_i$'s blockchain address and price $(\mathsf{addr_i}, price_i)$ associated with the parking owner P_i will be added to a new candidate set RS.

Finally, $\mathcal{C}$ sends a blockchain transaction that contains the parking spot matching result res to D_j with the output address $\mathsf{addr_j}$, where $res \in RS$ is the matched result with the lowest price. The driver D_j can contact to the parking owner associated with the address $\mathsf{addr_i}$, pay parking fees, and finally finish parking.

VI. Achieving Fairness

It is extremely important that the proposed system is not only privacy-preserving but also fair. In this regard, we present a complete analysis to prove that BSFP achieves fairness. In the following sub-sections, we focus on the design of the smart contract $\mathcal{C}$, the reward for honest workers, and the punishment of malicious workers and users.

A. Notations for Fairness Realization

Table II lists the notations used in the process of achieving fairness in BSFP.

TABLE II: List of Notations for Fairness Realization.

Notation	Description
addr_i	Parking owner P_i's blockchain address.
addr_j	Driver D_j's blockchain address.
$\mathsf{deposit}$	Initial deposited digital currency.
$\mathsf{balance}_\mathsf{Pi}$, $\mathsf{balance}_\mathsf{Dj}$	P_i and D_j's account balance.
$\mathsf{cost}_\mathsf{max}$	Cost limit of invoking $\mathcal{C}$.
cost_I	Contract fee of calling $\mathsf{SpotPub}()$ function.
cost_M	Contract fee of calling $\mathsf{Match}()$ function.
cost_P	Contract fee of calling $\mathsf{Payment}()$ function.
$\mathsf{payment}_\mathsf{ji}$	Advanced payment by D_j to P_i.

Smart contract $\mathcal{C}$

1. Contract Initialization: $\mathsf{Init}()$
1). TA deploys $\mathcal{C}$ to the blockchain.
2). Each parking owner P_i puts $\mathsf{deposit}$ to addr_i.
3). Each driver D_j puts $\mathsf{deposit}$ to addr_j.

2. Parking Spot Publishing: $\mathsf{SpotPub}(\mathcal{L}_i)$
1). Check $\mathsf{balance}_\mathsf{Pi} > \mathsf{cost}_\mathsf{max}$.
2). Perform *Time Verification*.
3). Perform *Parking Owner Authentication*.
4). Store $\mathcal{L}_i$ to blockchain.
5). Set $\mathsf{balance}_\mathsf{Pi} = \mathsf{balance}_\mathsf{Pi} - \mathsf{cost}_\mathsf{I}$.

3. Parking Spot Matching: $\mathsf{Match}(\mathcal{R}_j)$
1). Check $\mathsf{balance}_\mathsf{Dj} > \mathsf{cost}_\mathsf{max}$.
2). Perform *Parking Spot Matching*.
3). Set $\mathsf{balance}_\mathsf{Dj} = \mathsf{balance}_\mathsf{Dj} - \mathsf{cost}_\mathsf{M}$.
4). Return matching result res to D_j by transaction.

4. Payment: $\mathsf{Payment}(\mathsf{addr}_\mathsf{i}, \mathsf{payment}_\mathsf{ji})$
1). Check $\mathsf{balance}_\mathsf{Dj} > \mathsf{cost}_\mathsf{max} + \mathsf{payment}_\mathsf{ji}$.
2). Update $\mathcal{L}_i$'s available time vector.
3). Set $\mathsf{balance}_\mathsf{Dj} = \mathsf{balance}_\mathsf{Dj} - \mathsf{cost}_\mathsf{P} - \mathsf{payment}_\mathsf{ji}$.
4). Set $\mathsf{balance}_\mathsf{Pi} = \mathsf{balance}_\mathsf{Pi} + \mathsf{payment}_\mathsf{ji}$.

Fig. 3: Design of smart contract $\mathcal{C}$ for fairness.

Note that the $\mathsf{cost}_\mathsf{max}$ is considered as a fixed cost limit. For any contract function $\mathsf{F}()$, if the execution cost exceeds this limit, the execution of $\mathsf{F}()$ will be terminated, and the cost will not be returned to the caller. The cost_I, cost_M and cost_P represent the contract fee of calling $\mathsf{SpotPub}()$, $\mathsf{Match}()$ and $\mathsf{Payment}()$, respectively. The functional cost can be set as a fixed value or be related to the computation complexity of the function. In addition, the transaction fee is also considered, which is related to the data size stored on the transaction.

For example, the functional cost (i.e., contract fee) and the transaction fee can be quantified as $\mathsf{c}_\mathsf{F} \cdot \mathsf{fee}_\mathsf{comp}$ and $\mathsf{len}_\mathsf{T} \cdot \mathsf{fee}_\mathsf{data}$, respectively. More specifically, c_F represents the computational cost of function F, $\mathsf{fee}_\mathsf{comp}$ represents the fee of each consumed computational resource unit, len_T represents the byte length of transaction T, and $\mathsf{fee}_\mathsf{data}$ represents the fee of each stored byte. The set of cost is similar to the gas mechanism of Ethereum that uses Gas associated to its naive currency ETH to quantify the functional cost, and uses $\mathsf{gasLimit}$ to set the cost limit of a given smart contract.

B. Design of the Smart Contract

The smart contract $\mathcal{C}$ in the proposed scheme plays an integral part in achieving fairness and comprises four main functions as shown in Fig. 3. Each of these is discussed in detail below.

1) Contract Initialization: The function $\mathsf{Init}()$ is used to initialize the smart contract. First TA deploys $\mathcal{C}$ to the blockchain, and all the registered parking owners & drivers need to put $\mathsf{deposit}$ amounts of blockchain-based digital currency (e.g., ETH in Ethereum) to their blockchain address.

2) Parking Spot Publishing: The function $\mathsf{SpotPub}()$ is called by P_i to publish the parking spot information $\mathcal{L}_i$. First, the function checks whether P_i can afford the contract fees (that will be given to blockchain miners) by comparing $\mathsf{balance}_\mathsf{Pi}$ and $\mathsf{cost}_\mathsf{max}$. If sufficient balance is available to pay the fees, it will perform time verification and authentication as shown in Fig. 3. If the verification and authentication are successful, it will store $\mathcal{L}_i$ to blockchain and deduct the contract fee cost_I from the balance of P_i's address addr_i.

3) Parking spot Matching: The function $\mathsf{Match}()$ is called by D_j to find the parking spot that satisfies both location and time requirements. First the function checks D_j's balance that is similar to what $\mathsf{SpotPub}()$ does. If the checking is passed, it will perform parking spot matching as is given above. Because RSU undertakes the task of driver's authentication and verification, this function does not need to perform these redundant procedures. After matching, it deducts the contract fee cost_M from the balance of D_j's address addr_j.

4) Payment: The function $\mathsf{Payment}()$ is called by D_j to make an advance payment to the parking owner P_i. First the function checks whether D_j can afford both the contract fee and the payment by comparing $\mathsf{balance}_\mathsf{Dj}$ and $\mathsf{cost}_\mathsf{max} + \mathsf{payment}_\mathsf{ji}$. If the checking is successful, it will update $\mathcal{L}_i$'s available time vector. Then, it deducts the contract fee cost_P and pays $\mathsf{payment}_\mathsf{ji}$ to P_i from addr_j's balance.

C. How to Achieve Fairness

The proposed BSFP achieves fairness on all the entities (i.e., drivers, parking owners, and workers) who participated in the system.

1) Fair Payment: A major aspect of fairness is that both the driver and the parking owner legally receive what they are paying for. For a driver D_j, if he pays the fee, he will get the correct parking spot matching result. For a parking owner P_i, if he pays the fee, his parking spot information will be published. Moreover, if a driver rents a parking spot, the corresponding parking owner will get the payment sent by the driver. This kind of fairness can be practically guaranteed by the smart contract $\mathcal{C}$ designed in our proposal.

2) Rewards & Punishment: The second aspect of fairness is shown through the reward to an honest worker (i.e., the blockchain miner), while the misbehaving or malicious users are punished. This kind of fairness can be guaranteed by the blockchain network and TA's accountability. For honest workers, they will receive the mining rewards which consist of transaction fees and contract fees. For malicious users, all their interactions will be stored on the blockchain publicly and permanently. In case of any dispute, TA can trace the malicious users by checking its BBS signature in their parking spot information or parking requirements stored on the blockchain. Once the real identity of the malicious user is traced, TA will punish them in both economic and administrative terms.

D. Summary

In traditional smart parking, the centralized cloud server performs parking spot publishing for parking owners and performs parking spot matching for drivers. Unfortunately, a compromised cloud may drop the valid request for both drivers and parking owners, or return an inaccurate query result to drivers.

The decentralized structure of BSFP eliminates the malicious central node, realizes reliable parking spot publishing & matching, and also enhances the fairness of BSFP. Besides, the specialized design of the smart contract $\mathcal{C}$ solves the financial issues of smart parking in trustless circumstances. In general, BSFP achieves fairness for both parking owners and drivers. Meanwhile, since miner nodes' real identities can be traced, the honest miner nodes will be rewarded, while the misbehaving users will be punished.

VII. Security Analysis

In this section, we present a thorough security analysis to prove that BSFP achieves privacy protection, integrity, reliability, authentication, and traceability.

A. Identity Privacy Protection

We utilize the BBS group signature [42] to protect identity privacy. In BSFP, all the drivers belong to the same group and all the parking owners belong to another group. When the smart contract verifies the BBS signature σ, the only information that can be inferred is that the requester is a driver or a parking owner. In addition, adversaries cannot infer which driver or parking owner signs σ because the BBS signature achieves full-anonymity which ensures that σ cannot reveal the signer's identity.

Moreover, both drivers and parking owners use their blockchain address addr to invoke the smart contract or to communicate with others. Blockchain address can be seen as a pseudonym generated by themselves, and the generation of blockchain address is irrelevant to the owner's real identity. Hence, the identity privacy for both drivers and parking owners can be protected by the unlinkability of the BBS group signature and blockchain address.

B. Location Privacy Protection

When a parking owner P_i wants to publish their parking spot information, he converts the exact location to grid coordinates and obfuscates it by using pseudo-random functions F, G and Bloom Filters (BFs). The security of location encryption for location-based range query equals to proving that 1) the original data obfuscated and stored in the BF cannot be revealed, and 2) any two BFs are indistinguishable to any PPT adversaries. These two requirements can be guaranteed if the one-way pseudo-random functions F and G are secure.

When a driver D_j needs to find a parking spot, first he generates a query token of the queried grid range. The security of token generation for location-based range query equals to proving that the queried range cannot be revealed by query token. This requirement can be guaranteed if G is secure.

Hence, both driver's queried location privacy and parking owner's parking location privacy are protected by the cryptographic primitives mentioned above.

C. Time Privacy Protection

For a parking owner P_i, the available time vector α_i will be encrypted by $\alpha_i' = \alpha_i \cdot M_1$ in parking spot publishing phase. For a driver D_j, the parking time vector β_j will be encrypted by $\beta_j' = M_2 \cdot \beta_j^T$ in request generation phases. Without knowing M_1 and M_2, α_i and β_j cannot be recovered from α_i' and β_j', respectively. Besides, without the RSU R that holds the key $M_1^{-1} \cdot M_2^{-1}$, all the encrypted available time vector stored on the blockchain cannot be recovered by malicious driver that holds M_1. Hence the privacy of parking owner's available time and driver's parking time is protected.

D. Data Privacy Protection

The data stored on the blockchain mainly includes three pieces of information: parking spot information, parking requests, and the intermediate data of parking spot matching. Based on the above analysis, all the data stored on the blockchain is encrypted or obfuscated. Moreover, all the parking spot matching processes are based on ciphertexts which means that the blockchain does not need to decrypt them during the parking spot matching process. Hence, if all the cryptographic primitives that we use in BSFP are secure, the data privacy will be well protected.

E. Integrity & Reliability

We utilize hash function and digital signatures to realize the integrity during data transmission. In BSFP, each parking spot information and each parking request contains a BBS signature of the hashed data. Hence, if the BBS signature is secure, and the hash function H is collision-resistant, the data modification by malicious adversaries will be detected by the smart contract $\mathcal{C}$ or RSU when verifying the BBS signature. The blockchain transaction also uses the secure digital signature (e.g., ECDSA in Bitcoin and Ethereum), hence, the integrity during transaction transmission can also be guaranteed.

Blockchain is a reliable and decentralized network with immutability, hence the data stored on the blockchain cannot be tampered with by malicious adversaries. Moreover, the flooding of fake parking requests to congest the system cannot be realized because each parking request invokes the smart contract $\mathcal{C}$ which will cost at most $\mathsf{cost}_{\mathsf{max}}$ in contract fees. To send numerous parking requests, adversaries will have to spend a lot of money which will be unacceptable for them. Hence, reliable data storage can also be guaranteed.

F. Authentication & Traceability

Both the smart contract $\mathcal{C}$ and RSUs have to conduct time verification and authentication before handling the requests of parking owners and drivers. The time verification is used to resist replay attacks during authentication, and the BBS group signature is used for anonymous authentication. For registered users, they can use their private key provided by TA to generate a BBS signature for authentication. For unauthorized users, the BBS signature cannot be generated without having a private key. Hence, unauthorized users cannot impersonate authorized users to attack the whole system.

Although both drivers and parking owners use group signatures to achieve anonymous authentication, the identity of misbehaving users can be traced by TA. For instance, assume a parking owner P_i publishes a fake parking spot information $\mathcal{L}_i = (A'_i, \mathcal{B}^i_x, \mathcal{B}^i_y, r^i_x, r^i_y, \alpha'_i, \textsf{addr}_\textsf{i}, t_i, \sigma_i)$, then the TA can use the BBS signature σ_i to trace P_i's real-world identity by executing BBS.Open$(Ppk, Pmsk, H(tmp), \sigma_i)$, where $tmp = A'_i||\mathcal{B}^i_x||\mathcal{B}^i_y||r^i_x||r^i_y||\alpha'_i||\textsf{addr}_\textsf{i}||t_i$. Similarly, for misbehaved driver D_j, his real-world identity can also be traced when TA checks the BBS signature σ_j. Hence, traceability is also guaranteed.

VIII. Experimental Evaluation

To evaluate the quantitative metrics of the proposed BSFP scheme, we use the real-world datasets for extensive experimentation. Experimental results show that BSFP has high efficiency, which indicates that the proposed scheme is practical in real-world smart parking scenarios.

A. Experimental Setup

We implement the proposed BSFP scheme and deploy it in Hyperledger Fabric[1] blockchain platform, as it has flexible programmability and relatively high efficiency compared to other existing blockchain platforms. For the implementation of the anonymous group signature, we utilize the PBC Library[2,3]. All programs are written in Golang and compiled on Go 1.13.6 x64 environment.

We have used the T-Drive trajectory data sample[4] which includes the real-world trajectory data generated by 30,000 taxis in Beijing as a reference for real-world parking process. We also use the map of Downtown Beijing[5] to locate both the position (i.e., the latitude and longitude coordinates) of drivers and parking spots. These real-world datasets are preprocessed using the following steps: 1) We have selected the location data with latitude ranging from $39.8°$N to $40.0°$N and longitude ranging from $116.2°$E to $116.4°$E. 2) We then partition the selected region into 25 areas. Each area consists of $4 \times 4 = 16$ grids with the corresponding coordinates range from $(0, 0)$ to $(3, 3)$. 3) We randomly select 5000 locations for drivers and 2500 locations for parking owners. 4) For each driver, we calculate the current Area No. and randomly create the queried range based on their exact locations. The request time vector is randomly generated and follows a uniform distribution. 5) For each parking owner, we convert the exact location of the parking spot to Area No. and grid coordinates. The available time vector is randomly generated and follows a uniform distribution.

We set the security parameter $\lambda = 256$, the number of shared keys for location-based range query $o = 5$, the dimension of time vector and matrix $l = 12$, and the request time threshold $t_\theta = 120$s. The price for each parking spot is randomly generated which follows a uniform distribution ranging from 1 to 10. We use SHA256 and HMAC to instantiate the hash function H and the pseudo-random function F, G, respectively. We use the RSA cryptosystem with 2048bit key length to realize the signature generation & verification for R and $\mathcal{C}$, respectively.

In order to execute the smart contract in BSFP, we deploy a test blockchain network consisting of four peer nodes running Hyperledger Fabric 1.3. Each blockchain node is running on a server with an Intel i7-9700K processor with 16G RAM and 64-bit Ubuntu 16.04 operating system. The clients (for RSU, driver, and parking owner) of BSFP are running on a system with an Intel i5-7300U processor with 8GB RAM and 64-bit Ubuntu 16.04 operating system.

TABLE III: Experimental Scenarios.

Scenario	Number of Drivers	Number of Parking Spot Owners
1	500	250
2	1000	500
3	1500	750
4	2000	1000
5	2500	1250
6	3000	1500
7	3500	1750
8	4000	2000
9	4500	2250
10	5000	2500

We simulate 10 different scenarios to analyze the influence of the varying number of drivers and available parking spots. As is shown in Table III, Scenario 1 and Scenario 10 have the minimum and the maximum number of drivers and parking spot owners, respectively. In our experimental scenario, each parking owner publishes one parking spot to the blockchain, and each driver sends one parking request to the blockchain. We run the simulation for each scenario 5 times and compute the average execution time to reduce errors.

B. Time Complexity

We present a detailed time complexity analysis of the whole smart parking process to prove that the proposed BSFP scheme has high computation efficiency.

To analyze the time costs quantitatively, we use several notations that represent the number of entities and the time costs of some basic operations. The list of notations is shown in Table IV. Note that the execution time of some operations (e.g., integer comparison in time verification process) is negligible, hence we neglect the time cost of these operations.

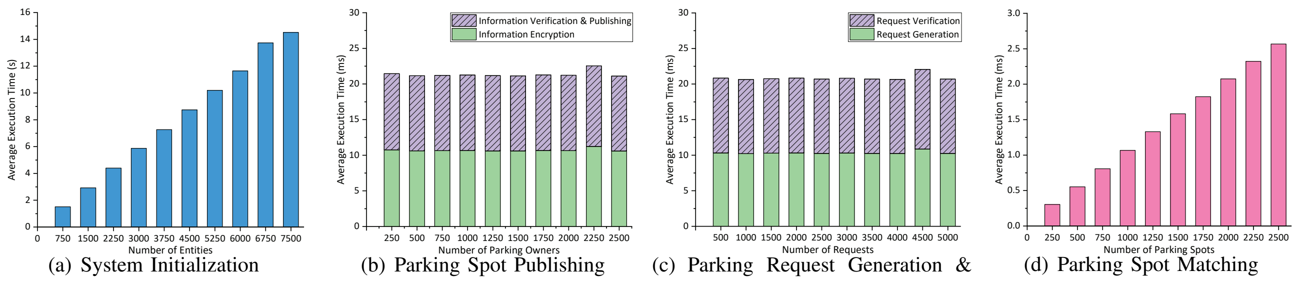
[1]https://www.hyperledger.org/projects/fabric

[2]https://crypto.stanford.edu/pbc/

[3]https://github.com/Nik-U/pbc

[4]https://www.microsoft.com/en-us/research/publication/t-drive-trajectory-data-sample/

[5]http://www.chinatouristmaps.com/travel/beijing/street-map.html



(a) System Initialization (b) Parking Spot Publishing (c) Parking Request Generation & Verification (d) Parking Spot Matching

Fig. 4: Time Performance of BSFP.

TABLE IV: Notations for Time Complexity Analysis.

Notation	Description
m	The number of parking owners.
n	The number of drivers.
t_{reg}	Time for entity registration.
t_{sig}	Time for BBS signature generation.
t_{ver}	Time for BBS signature verification.
t_{mul}	Time for matrix multiplication.
t_{lenc}	Time for encrypting parking owner's exact location.
t_{token}	Time for generating location-based range search token.
t_{query}	Time for executing location-based querying.
t_{rsig}	Time for RSA signature generation.
t_{rver}	Time for RSA signature verification.
t_{comm}	Communication delay for one side.

1) System Initialization: In the initialization process, TA will generate and distribute the shared keys, the shared matrices, and the BBS group private key as the registration information for each entity. Hence, the execution time of system initialization is approximately $(m+n)\cdot t_{reg}$, and the corresponding time complexity is $O(m+n)$.

It can be observed from Fig. 4(a) that as the total number of entities (i.e., $m+n$ for each scenario) increases, the execution time also increases, which is quite intuitive. On average, each registration process only costs about 1.93ms. Consequently, the time cost of system initialization is dependent on the number of registered entities, and it is negligible in practical scenarios.

2) Parking Spot Publishing: After initialization, the parking owner will generate parking spot information which consists of an obfuscated location, an encrypted time vector, an address, and a BBS signature. Then it invokes the smart contract $\mathcal{C}$ to publish the parking spot information to the blockchain. Hence, for each parking owner, the execution time of parking spot information encryption is approximately $t_{lenc}+t_{mul}+t_{sig}$. For $\mathcal{C}$, the execution time of verification is about t_{ver}. Therefore, the execution time of parking spot publishing is about $t_{lenc}+t_{mul}+t_{sig}+t_{ver}+t_{comm}$, and the corresponding time complexity is $O(1)$. Note that the communication delay t_{comm} is highly dependent on the network topological structure, interface & link bandwidth, and the Fabric itself. Hence, we omit the analysis of communication delay because its optimization is beyond the scope of this work. In the proposed scheme's implementation, t_{comm} was observed to be approximately 40ms, which is acceptable.

Fig. 4(b) shows the average execution time cost for encryption for each driver and verification & publishing for the smart contract (*chaincode*). It can also be observed in Fig. 4(b) that the time cost for encryption, validation, and publishing are not dependent on the number of parking owners. More specifically, it costs approximately 10.59ms for a parking owner to encrypt the parking spot information, while the verification & publishing time is about 10.52ms. Hence, it can be concluded that the proposed scheme is efficient and highly scalable.

3) Parking Request Generation: When a driver wants to find a suitable parking spot, it first generates a parking request that consists of an obfuscated location search token, an encrypted time vector, and a BBS signature, and sends it to the nearby RSU. Hence, for each parking owner, the execution time of request generation is computed as $t_{token}+t_{mul}+t_{sig}$. Then the RSU verifies the request that includes time verification (of which the time cost is negligible), BBS signature verification. If the verification is successful, it re-encrypts the request time vector, generates an RSA signature, and forwards the modified request to the blockchain. Hence, for RSU, the execution time of request verification and new request generation is $t_{ver}+t_{mul}+t_{rsig}$. Therefore, the execution time of request sending is $t_{token}+2t_{mul}+t_{sig}+t_{ver}+t_{rsig}+t_{comm}$, and the time complexity of request sending is $O(1)$.

It can be observed in Fig. 4(c) that these two steps are efficient and independent of the number of drivers in the system. More specifically, the average execution times for request generation and verification are only 10.25ms and 10.46ms, respectively.

4) Parking Spot Matching: During the parking spot matching process, $\mathcal{C}$ first verifies RSU's signature, then all the parking spots in the driver's area will be matched to get the candidate matching results by executing at most m querying (for location-based matching) operations and m matrix multiplication (for time-based matching) operations. Hence, the execution time of request matching is $t_{rver}+m\cdot(t_{query}+t_{mul})$, and the corresponding time complexity is $O(m)$.

Fig. 4(d) shows that when the number of the parking owners increases, the average execution time also increases. As there are more parking spots available, the time required to match and find suitable ones also increases proportionally.

TABLE V: Communication Costs of BSFP.

Process	Communication Costs
Parking Spot Publishing	8.47KB
Parking Request Generation	2.40KB
Parking Spot Matching	0.11KB

C. Communication Complexity

In this set of experiments, we analyze the communication complexity of BSFP, which represents the number of bits exchanged between those entities, to prove that BSFP has high communication efficiency.

In the *Parking Spot Publishing* process, the encrypted parking spot information will be sent and published on the blockchain. Therefore, the communication complexity equals to the size of the blockchain transaction that stores the parking owner P_i's encrypted parking spot $\mathcal{L}_i$. As is mentioned in Section III, each blockchain transaction consists of a transaction head and a transaction body. The size of a transaction head is determined by the blockchain itself, we use $size(head)$ as the size of a transaction head for simplicity, where the function of $size(a)$ represents the bit length of a. For the transaction body, it contains the stored parking spot information $\mathcal{L}$ and several contract parameters for invoking the smart contract $\mathcal{C}$. The bit length of these contract parameters is far less than that of $\mathcal{L}_i$, hence the size of a transaction approximately equals $size(head) + size(\mathcal{L}_i)$. Consequently, the communication complexity between the parking owner and the blockchain in this process achieves $O(1)$.

In the *Parking Request Sending* process, the driver D_j sends the parking request $\mathcal{R}_j$ to a nearby RSU, following which the RSU verifies the request and forwards it to the blockchain. The communication cost of this process is $size(head) + size(\mathcal{R}_j)$. The corresponding communication complexity between the user and the RSU and that between the RSU and the blockchain are both $O(1)$.

In the *Parking Spot Matching* process, after matching the result res will be sent to the driver through RSUs. The communication cost of this process is $size(head)+size(res)$. The corresponding communication complexity between the blockchain and the RSU and that between the RSU and the blockchain are both $O(1)$.

Table V illustrates the overall communication costs of BSFP. The cost of parking spot matching is the highest. However, it only takes less than 8.5KB to store information about a parking spot. Moreover, in a real-world scenario, once a parking owner publishes their parking spots, only the available time will be changed, which only costs 0.19KB in our experiment. Hence the execution of parking spot publishing is not communication intensive. It should be noted that the communication cost of other processes is less than 2.5KB each, which makes the entire scheme significantly efficient and scalable.

D. Summary

In Table VI we summarize both the time and communication complexity of BSFP. It can be observed that BSFP achieves $O(1)$ time complexity in most processes and achieves $O(1)$ communication complexity in all processes. Experimental results based on real-world datasets show that it only costs several hundred milliseconds and less than 10KB for each driver and parking owners to complete the whole process of smart parking. Even in scenario 10 where 7500 entities participate in the system, the execution time is still efficient. In conclusion, BSFP achieves both computation and communication efficiency, and it can be used in real-world smart parking scenarios.

IX. Conclusion and Future Work

This work presents a novel system model for decentralized and reliable smart parking, where the blockchain is leveraged to guarantee data reliability. The proposed BSFP, a blockchain-based smart parking scheme with fairness, reliability and privacy protection, utilizes the BBS group signature, secure pseudo-random function, bloom filter, and vector-based encryption to protect the privacy for both drivers and parking owners while enabling the traceability of TA to find the misbehaving users. To achieve fairness, we design a smart contract that can guarantee that every honest user and worker in the system can get the correct results and the corresponding rewards, while the misbehaving or malicious users and workers are punished. A thorough security analysis proves that the proposed BSFP scheme protects identity privacy, location privacy, time privacy, and data privacy. It also proves that BSFP realizes integrity, reliability, authentication, and traceability. Comprehensive experiments based on real-world dataset shows that BSFP achieves computation and communication efficiency.

In the future, we intend to consider a more flexible matching strategy. Currently, BSFP considers location, time, and parking price as the main factors for parking spot matching. Moreover, the reputation score can also be used to evaluate the behavior of both drivers and parking owners. Achieving privacy-preserving reputation calculation, matching, and updates will be a challenging task in this regard. Another extension can focus on the use of fog nodes, which can further reduce both the computation and communication costs of the entire system. However, reliability and privacy issues for fog nodes, in addition to their integration with blockchain is also a challenge issue.

References

[1] (2017) Smart apps are solving china's parking problems. [Online]. Available: http://english.gov.cn/news/video/2017/09/25/content_281475882375396.htm

[2] D. C. Shoup, "Cruising for parking," *Transport Policy*, vol. 13, no. 6, pp. 479 – 486, 2006, parking.

[3] Y. Xue, K. Xue, N. Gai, J. Hong, D. S. L. Wei, and P. Hong, "An attribute-based controlled collaborative access control scheme for public cloud storage," *IEEE Trans. Information Forensics and Security*, vol. 14, no. 11, pp. 2927–2942, 2019.

[4] S. Xu, G. Yang, Y. Mu, and R. H. Deng, "Secure fine-grained access control and data sharing for dynamic groups in the cloud," *IEEE Trans. Information Forensics and Security*, vol. 13, no. 8, pp. 2101–2113, 2018.

[5] S. Xu, Y. Li, R. H. Deng, Y. Zhang, X. Luo, and X. Liu, "Lightweight and expressive fine-grained access control for healthcare internet-of-things," *IEEE Transactions on Cloud Computing*, 2019.

[6] C. Zhang, L. Zhu, C. Xu, K. Sharif, C. Zhang, and X. Liu, "PGAS: privacy-preserving graph encryption for accurate constrained shortest distance queries," *Inf. Sci.*, vol. 506, pp. 325–345, 2020.

TABLE VI: Time and Communication Complexity of BSFP.

Process	Time Complexity	Communication Complexity for Driver	Communication Complexity for Parking Owner
System Initialization	$O(m+n)$	-	-
Parking Spot Publishing	$O(1)$	-	$O(1)$
Parking Request Generation	$O(1)$	$O(1)$	-
Parking Spot Matching	$O(m)$	$O(1)$	-

[7] S. Xu, G. Yang, and Y. Mu, "Revocable attribute-based encryption with decryption key exposure resistance and ciphertext delegation," *Inf. Sci.*, vol. 479, pp. 116–134, 2019.

[8] X. Yang, R. Lu, J. Shao, X. Tang, and H. Yang, "An efficient and privacy-preserving disease risk prediction scheme for e-healthcare," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 3284–3297, 2019.

[9] K. Xue, S. Li, J. Hong, Y. Xue, N. Yu, and P. Hong, "Two-cloud secure database for numeric-related SQL range queries with privacy preserving," *IEEE Trans. Information Forensics and Security*, vol. 12, no. 7, pp. 1596–1608, 2017.

[10] J. Wang, J. Liu, and N. Kato, "Networking and communications in autonomous driving: A survey," *IEEE Communications Surveys and Tutorials*, vol. 21, no. 2, pp. 1243–1274, 2019.

[11] C. Zhang, L. Zhu, C. Xu, K. Sharif, and X. Liu, "PPTDS: A privacy-preserving truth discovery scheme in crowd sensing systems," *Inf. Sci.*, vol. 484, pp. 183–196, 2019.

[12] L. Zhu, C. Zhang, C. Xu, and K. Sharif, "Rtsense: Providing reliable trust-based crowdsensing services in CVCC," *IEEE Network*, vol. 32, no. 3, pp. 20–26, 2018.

[13] Q. Yuan, H. Zhou, Z. Liu, J. Li, F. Yang, and X. Shen, "Cesense: Cost-effective urban environment sensing in vehicular sensor networks," *IEEE Trans. Intelligent Transportation Systems*, vol. 20, no. 9, pp. 3235–3246, 2019.

[14] F. Lyu, H. Zhu, H. Zhou, W. Xu, N. Zhang, M. Li, and X. Shen, "SS-MAC: A novel time slot-sharing MAC for safety messages broadcasting in vanets," *IEEE Trans. Vehicular Technology*, vol. 67, no. 4, pp. 3586–3597, 2018.

[15] F. Lyu, H. Zhu, H. Zhou, L. P. Qian, W. Xu, M. Li, and X. Shen, "Momac: Mobility-aware and collision-avoidance MAC for safety applications in vanets," *IEEE Trans. Vehicular Technology*, vol. 67, no. 11, pp. 10 590–10 602, 2018.

[16] K. Xue, J. Hong, Y. Ma, D. S. L. Wei, P. Hong, and N. Yu, "Fog-aided verifiable privacy preserving access control for latency-sensitive data sharing in vehicular cloud computing," *IEEE Network*, vol. 32, no. 3, pp. 7–13, 2018.

[17] J. Shao and G. Wei, "Secure outsourced computation in connected vehicular cloud computing," *IEEE Network*, vol. 32, no. 3, pp. 36–41, 2018.

[18] L. Zhu, C. Zhang, C. Xu, X. Du, R. Xu, K. Sharif, and M. Guizani, "PRIF: A privacy-preserving interest-based forwarding scheme for social internet of vehicles," *CoRR*, vol. abs/1804.02440, 2018.

[19] Y. Li, Q. Luo, J. Liu, H. Guo, and N. Kato, "TSP security in intelligent and connected vehicles: Challenges and solutions," *IEEE Wireless Commun.*, vol. 26, no. 3, pp. 125–131, 2019.

[20] R. Lu, X. Lin, H. Zhu, and X. Shen, "An intelligent secure and privacy-preserving parking scheme through vehicular communications," *IEEE Trans. Vehicular Technology*, vol. 59, no. 6, pp. 2772–2785, 2010.

[21] J. Ni, K. Zhang, Y. Yu, X. Lin, and X. Shen, "Privacy-preserving smart parking navigation supporting efficient driving guidance retrieval," *IEEE Trans. Vehicular Technology*, vol. 67, no. 7, pp. 6504–6517, 2018.

[22] L. Zhu, M. Li, Z. Zhang, and Z. Qin, "Asap: An anonymous smart-parking and payment scheme in vehicular networks," *IEEE Transactions on Dependable and Secure Computing*, pp. 1–1, 2018.

[23] W. A. Amiri, M. Baza, K. Banawan, M. M. E. A. Mahmoud, W. Alasmary, and K. Akkaya, "Privacy-preserving smart parking system using blockchain and private information retrieval," *CoRR*, vol. abs/1904.09703, 2019.

[24] T. Lin, H. Rivano, and F. L. Mouel, "A survey of smart parking solutions," *IEEE Trans. Intelligent Transportation Systems*, vol. 18, no. 12, pp. 3229–3253, 2017.

[25] J. Lin, S. Chen, C. Chang, and G. Chen, "SPA: smart parking algorithm based on driver behavior and parking traffic predictions," *IEEE Access*, vol. 7, pp. 34 275–34 288, 2019.

[26] S. R. Rizvi, S. Zehra, and S. Olariu, "ASPIRE: an agent-oriented smart parking recommendation system for smart cities," *IEEE Intell. Transport. Syst. Mag.*, vol. 11, no. 4, pp. 48–61, 2019.

[27] S. Nakamoto. (2008) Bitcoin: A peer-to-peer electronic cash system. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[28] J. Liu, Y. Shi, Z. M. Fadlullah, and N. Kato, "Space-air-ground integrated network: A survey," *IEEE Communications Surveys and Tutorials*, vol. 20, no. 4, pp. 2714–2741, 2018.

[29] N. Kato, Z. M. Fadlullah, F. Tang, B. Mao, S. Tani, A. Okamura, and J. Liu, "Optimizing space-air-ground integrated networks by artificial intelligence," *IEEE Wireless Commun.*, vol. 26, no. 4, pp. 140–147, 2019.

[30] J. Li, H. Lu, K. Xue, and Y. Zhang, "Temporal netgrid model-based dynamic routing in large-scale small satellite networks," *IEEE Trans. Vehicular Technology*, vol. 68, no. 6, pp. 6009–6021, 2019.

[31] H. Zhou, N. Cheng, J. Wang, J. Chen, Q. Yu, and X. Shen, "Toward dynamic link utilization for efficient vehicular edge content distribution," *IEEE Transactions on Vehicular Technology*, 2019.

[32] S. Biswas, K. Sharif, F. Li, B. Nour, and Y. Wang, "A Scalable Blockchain Framework for Secure Transactions in IoT," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4650–4659, 2019.

[33] F. Gao, L. Zhu, M. Shen, K. Sharif, Z. Wan, and K. Ren, "A blockchain-based privacy-preserving payment mechanism for vehicle-to-grid networks," *IEEE Network*, vol. 32, no. 6, pp. 184–192, 2018.

[34] R. Sharma and S. Chakraborty, "B2VDM: blockchain based vehicular data management," in *2018 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2018, Bangalore, India, September 19-22, 2018*, 2018, pp. 2337–2343.

[35] M. Li, L. Zhu, and X. Lin, "Efficient and privacy-preserving carpooling using blockchain-assisted vehicular fog computing," *IEEE Internet of Things Journal*, pp. 1–1, 2019.

[36] I. Bentov and R. Kumaresan, "How to use bitcoin to design fair protocols," in *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part II*, 2014, pp. 421–439.

[37] M. Andrychowicz, S. Dziembowski, D. Malinowski, and L. Mazurek, "Secure multiparty computations on bitcoin," in *2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, May 18-21, 2014*, 2014, pp. 443–458.

[38] R. Kumaresan and I. Bentov, "How to use bitcoin to incentivize correct computations," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014*, 2014, pp. 30–41.

[39] A. E. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, "Hawk: The blockchain model of cryptography and privacy-preserving smart contracts," in *IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, May 22-26, 2016*, 2016, pp. 839–858.

[40] L. Li, J. Liu, L. Cheng, S. Qiu, W. Wang, X. Zhang, and Z. Zhang, "Creditcoin: A privacy-preserving blockchain-based incentive announcement network for communications of smart vehicles," *IEEE Trans. Intelligent Transportation Systems*, vol. 19, no. 7, pp. 2204–2220, 2018.

[41] S. Hu, C. Cai, Q. Wang, C. Wang, X. Luo, and K. Ren, "Searching an encrypted cloud meets blockchain: A decentralized, reliable and fair realization," in *2018 IEEE Conference on Computer Communications, INFOCOM 2018, Honolulu, HI, USA, April 16-19, 2018*, 2018, pp. 792–800.

[42] D. Boneh, X. Boyen, and H. Shacham, "Short group signatures," in *Advances in Cryptology - CRYPTO 2004, 24th Annual International CryptologyConference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, 2004, pp. 41–55.

[43] R. Li, A. X. Liu, A. L. Wang, and B. Bruhadeshwar, "Fast range query processing with strong privacy protection for cloud computing," *PVLDB*, vol. 7, no. 14, pp. 1953–1964, 2014.

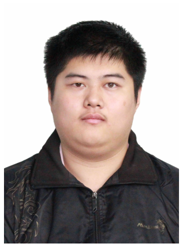
[44] G. Wood. (2014) Ethereum: A secure decentralized generalized transaction ledger. [Online]. Available: http://gavwood.com/paper.pdf

[45] N. Szabo. (1994) Smart contracts. [Online]. Available: http://szabo.best.vwh.net/smart.contracts.html

[46] J. Katz and Y. Lindell, *Introduction to Modern Cryptography, Second Edition*. CRC Press, 2014.

[47] A. Yang, J. Weng, N. Cheng, J. Ni, X. Lin, and X. Shen, "Deqos attack: Degrading quality of service in vanets and its mitigation," *IEEE Trans. Vehicular Technology*, vol. 68, no. 5, pp. 4834–4845, 2019.

[48] L. Wang, X. Shen, J. Li, J. Shao, and Y. Yang, "Cryptographic primitives in blockchains," *J. Network and Computer Applications*, vol. 127, pp. 43–58, 2019.



Can Zhang received his B.E. (Bachelor of Engineering) degree in Computer Science & Technology from Beijing Institute of Technology, Beijing, China, in 2017. He is currently a Ph.D. student at the School of Computer Science & Technology, Beijing Institute of Technology. His current research interests include security & privacy in VANET, cloud computing security, and blockchain technology.

Liehuang Zhu received his Ph.D. degree in computer science from Beijing Institute of Technology, Beijing, China, in 2004. He is currently a professor at the School of Computer Science and Technology, Beijing Institute of Technology. His research interests include security protocol analysis and design, group key exchange protocols, wireless sensor networks, cloud computing, and blockchain applications.

Chang Xu received her Ph.D. degree in computer science from Beihang University, Beijing, China, in 2013. She is currently an associate professor at the School of Computer Science and Technology, Beijing Institute of Technology. Her research interests include security & privacy in VANET, and big data security.

Chuan Zhang received his bachelor's degree in network engineering from Dalian University of Technology, Dalian, China, in 2015. He is currently a Ph.D. student at the School of Computer Science and Technology, Beijing Institute of Technology. His current research interests include secure data services in cloud computing, security & privacy in IoT, and big data security.

Kashif Sharif [M'08] received his MS degree in information technology in 2004, and PhD degree in computing and informatics from University of North Carolina at Charlotte, USA in 2012. He is currently an associate professor at Beijing Institute of Technology, China. His research interests include information centric networks, blockchain & distributed ledger technologies, wireless & sensor networks, software defined networks, and data center networking. He also serves as associate editor for IEEE Access.

Huishu Wu received his Bachelor degree in Information management of Computer Science in Hebei Normal University, and received the Master degree in management in China University of Political Science and Law (CUPL) and the Master degree in University of Montreal (UDEM). Since 2017, he pursues his doctoral study in University of Montreal. His research interest is in the area of data security & privacy in VANETs, and data governance.

Hannes Westermann is a Technologist. He has always had a strong passion for technology. He is also a researcher at the Cyberjustice Laboratory where he is leading the JusticeBot project, which aims to predict court case outcomes using artificial intelligence. His research interest is in the area of data analysis, data security, and data governance.